Simulation studies of Gigabit ethernet versus Myrinet using real application cores

Helen Chen
Sandia National Laboratories, MS 9011
P.O. Box 969, Livermore, CA 94551, USA
Phone 925 294 2991 FAX 925 294 1225
hycsw@ca.sandia.gov

Pete Wyckoff*
Sandia National Laboratories, MS 9011
P.O. Box 969, Livermore, CA 94551, USA
Phone 925 294 3503 FAX 925 294 1225
wyckoff@ca.sandia.gov

Presented at CANPC'00 workshop of High-Performance Computer Architecture, Toulouse, France, January 2000.

Keywords: simulation, gigabit ethernet, myrinet, topology.

Abstract

Parallel cluster computing projects use a large number of commodity PCs to provide cost-effective computational power to run parallel applications. Because properly load-balanced distributed parallel applications tend to send messages synchronously, minimizing blocking is as crucial a requirement for the network fabric as are those of high bandwidth and low latency. We consider the selection of an optimal, commodity-based, interconnect network technology and topology to provide high bandwidth, low latency, and reliable delivery.

Since our network design goal is to facilitate the performance of real applications, we evaluated the performance of myrinet and gigabit ethernet technologies in the context of working algorithms using modeling and simulation tools developed for this work.

Our simulation results show that myrinet behaves well in the absence of congestion. Under heavy load, its latency suffers due to blocking in the distributed wormhole routing scheme.

Conventional gigabit ethernet switches can not scale to support more than 64 gigabit ethernet ports today which leads to the use of cascaded switches. Bandwidth limitation in the interswitch links and extra store-and-forward delays limit aggregate performance of this configuration.

The Avici switch router uses six 40 Gbps internal links to connect individual switching nodes in a wormhole-routed three-dimensional torus. Additionaly, the fabric's large speed-up factor and its per-connection buffer management scheme provides for non-blocking deliveries under heavy load.

1 Introduction

A group of enthusiasts of commodity high-performance computing platforms has lived at the fringes of the computing community for many years now, and this bunch is seeing its numbers grow following the general demise of the massively parallel processor (MPP) manufacturers. Even the government research laboratories are joining the fray. Sandia National Laboratories is a United States Department of Energy research facility which can no longer satisfy its thirst for FLOPS by buying monolithic multi-million dollar machines, as there is not sufficient market demand to keep vendors in business.

The idea of cluster computing is to aggregate machine rooms full of relatively cheap hardware, connected with some sort of network, and apply the combined force of the individual machines on a single calculation. Problems arise, though, in attempting to operate this set of machines as a single unit. As it is not feasible to run a single instance of the operating system on the entire cluster, the alternate paradigm of message passing is used instead. Each processor (which could also be a small shared-memory multiprocessor) maintains a disjoint address space, and messages are passed between machines as driven by the requirements of each application.

The hardware employed in a cluster is generally the most readily available in the volume personal computing market, so as to leverage the cost advantages of buying commodity hardware. The down side to this is that some critical pieces of hardware for cluster computing are completely irrelevant for the mass market, namely the interconnect. The advent of shared 10 Mb/s ethernet [3] was a giant step, and remains the basis of the standard "fast" networking infrastructure, as it has been for the last 15 years. Within the last few years, though, the price of 100 Mb/s ethernet cards have approached the reach of most users, and commodity gigabit network components are on their way. High-end cluster users can afford both gigabit ethernet and myrinet [9] as the message passing infrastructure. More esoteric networking components, such as HiPPi [10] and Giganet [12] are

* Corresponding author

available to those willing to incur the additional costs, and are becoming cheaper and faster with time.

The remainder of this paper discusses the technologies we simulated, and methods we used which involve a mix of "artificial" basic tests and simulations of core algorithms from real parallel applications. Our results tally the positive and negative aspects of each technology.

2 Interconnect technologies

The following three subsections describe the network fabrics we considered in the simulations discussed in Section 3. In each section we calculate the current pricing for a prototypical 256 node cluster, a size which should be familiar to many cluster builders.

2.1 Myrinet

Myricom's myrinet is a cost-effective, high-performance communication and switching technology. It interconnects hosts and switches using 1.28 Gb/s full-duplex links. The myrinet PCI host adapter can be programmed to interact directly with the host processors for low-latency communications, and with the network to send, receive, and buffer packets.

Myricom supplies open source software that runs on common hosts and operating systems. This software maps the network periodically to find available paths between communicating hosts. All myrinet packets carry a source-based routing header to provide intermediate switches with forwarding directions. Therefore, myrinet switches do not need to run routing algorithms or maintain a routing table. Because myrinet does not impose a size limitation on its packets, it can easily encapsulate any protocol's packet format (e.g. TCP [8], IP, etc.), thereby providing interoperability. While the simplicity in the myrinet switch offers a low per-port cost, it lacks management capability to maintain robustness in large clusters.

The current myrinet switch is a 16-port crossbar, although there should soon be available a 64-port switch. These ports can be used to interconnect either switches or processors, thereby allowing arbitrary network topologies. Normally, more interswitch connections implies more diverse paths, which can reduce blocking within the switching fabric. However, there will then be fewer ports available to interconnect processing nodes.

The severe cable length restriction is the greatest impediment to creating complex topologies. Optical converters are available from Myricom, at a cost of $3600 per connection, which would more than double the per-host connection cost. On the small scale, one can easily build hypercubes and large-dimensional tori using 35 foot LAN cables. For our large scale simulations, we chose a two dimensional torus as the best tradeoff in terms of area and cost. It scales in two physical dimensions just as our hardware scales in two dimensions on the machine-room floor. Our plans for 10 000 compute nodes and our budget do not permit a hypercube topology on that scale.

Myrinet sells a network interface card for $1700, 16-port switches for $5000, and cables for $200. For the topology described above, the total cost for 256 nodes is $256 \times \$1700 + 32 \times \$5000 + 12 \times 32 \times \$200 = \$670$k.

2.2 Gigabit ethernet

The most popular Local Area Network (LAN) technology is ethernet. Ethernet has evolved from the 3 Mb/s technology, invented by Bob Metcalfe in 1973, to the 10-, 100- (or fast), and 1000-Mb/s (or gigabit) ethernet standards of today [7]. Riding on the ethernet popularity current, gigabit ethernet is fast becoming a commodity item and therefore, we believe it can be a cost-effective alternative to interconnect parallel computers. Moreover, there are already discussions of 10- and even 100-gigabit per second ethernet [4], which could provide the next generation parallel computers with a smooth upgrade path to their communication subsystem.

Conventional routers, however, are not scalable because they use designs based on a backplane bus or crossbar switch. The largest non-blocking switch available today supports only 64 nodes, and cascading is required to build a cluster beyond that size. These routers use the spanning tree algorithm to calculate a loop-free tree that has only a single path for each destination, using the redundant paths as hot stand-by links, precluding the use of, say, a mesh topology. Without diverse paths, cascaded switches will suffer performance bottlenecks due to output port contention.

Due to the lack of switch scalability and the necessity to remain backward compatible with slower ethernet implementations, we believe the applications of a conventional gigabit ethernet switch fabric are limited to small parallel systems. We decided to conduct a simulation study of a 256-node cluster, nevertheless, in order to evaluate the effects of ethernet's packet framing, inter-frame gap, maximum and minimum packet size, and store-and-forward switching mechanism on the performance of parallel applications.

Network cards for gigabit ethernet are around $700, and 64-port switches can be found for $30k. Including fiber and thinking forward to inter-switch trunking gives a total 256-node cost of $256 \times \$700 + 5 \times \$30\,000 + 280 \times \$75 = \$350\text{k}$.

2.3 Avici terabit switch router

The Avici router uses two direct-connect networks [2] as its switching fabric to achieve high performance, economical scalability, and robustness. The dual fabric connects switching nodes (or line cards) using twelve 20-Gbps full duplex links to form two 3-D toroidal meshes [1]. Each set of five line cards is grouped into a quadrant which is connected via a backplane to form a loop in the z-dimension. The x and y dimensions are formed by connecting neighboring quadrants along the backplane in folded tori, which allows uniformly short wires to be used for all connections, thereby lowering wiring costs as well as latency variations. With this arrangement, an Avici router can be incrementally expanded to include up to $14 \times 16 \times 5 = 1120$ line cards. At 16 gigabit ethernet ports per line cards, this configuration can interconnect a parallel system of 17 920 compute processors.

Similar to myrinet, the Avici router uses wormhole routing inside the fabric to achieve low latency. Unlike myrinet, however, rather than buffering the entire message inside the network, the Avici router segments its messages into 72-byte scheduling units and exercises credit-based flow control to prevent flit loss. Together with its per-connection buffer management, and over-provisioned fabric links relative to the line card I/O demands, the Avici router implements an output-buffered virtual crossbar to eliminate the blocking problem in wormhole routing. Because of the huge speed mismatch between gigabit ethernet and the fabric link (1:20), the Avici router will store incoming gigabit ethernet packets before forwarding to prevent buffer underrun within the fabric. Unlike conventional switches, however, the number of store-and-forward operations in the Avici reaches an upper bound of two, once at the incoming and the other at the outgoing gigabit ethernet port. Moreover, because the Avici router is designed for telecommunications applications, it is extremely robust and has extensive SNMP-based management capabilities, a feature that is essential to building reliable large parallel systems.

Replacing the switches in the cost calculation for conventional gigabit ethernet, and dropping a few unneeded fibers gives a total $256 \times \$700 + \$250\,000 + 256 \times \$75 = \$450\text{k}$.

3 Simulation methodologies

We adapt existing simulation packages to capture important characteristics of a technology, such as its link level protocol and switch architecture. Because these characteristics are unique to the technology it represents, we are not concerned with effects due to differences in implementation. Instead, we ensure the fidelity of our simulation results by extending these packages to use the same set of parallel algorithms to generate traffic. We also code an identical interface layer to handle details of packet transmission and reception. Since the goal of our study is to identify potential interconnect technologies, we do not consider end station overhead. We plan to address the performance issues involving the host adapter, device driver, and end-system protocol processing in a future study.

3.1 Opnet

MIL3's Optimized Engineering Tools [11] is a comprehensive engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis. Its features include graphical specification of models, event-scheduled simulation kernel, and hierarchical object-based modeling. We selected Opnet to simulate the conventional gigabit ethernet switch because Opnet has an existing model that simulates the gigabit ethernet protocol.

On the top level, we used Opnet's network editor to compose our network using components such as switches, nodes, and links. The network consists of five conventional gigabit ethernet switches, 256 compute nodes, and full-duplex links to interconnect them. We populated four switches each with 64 end nodes, which are in turn connected via a fifth switch. We chose a star topology because it offers the lowest hop count between the most distant nodes in the network.

At the next lower level, we used Opnet's node editor to construct our compute and switch nodes. A compute node consists of a module to run the parallel application models that we wrote, a gigabit ethernet protocol entity, a transmitter, and a receiver. The process model contains a state transition diagram which represents the parallel code. Each of the 256 nodes in the system runs the state machine which transitions between sending, receiving, and computing states.

3.2 Avici simulator

Allen King and coworkers at Avici wrote a simulator [5] to be used in planning the switch hardware they built. We inserted hooks into the simulator by which we could feed our own traffic patterns into the switch: `sendPacket()` inserts a packet into the fabric, while `receivedPacket()` is called up from the fabric to notify our modules of the receipt of a packet at a destination line card. The function `nextFlitTime()` notifies our code that the simulator has advanced in time, and we use that notification to fire any pending events. Various other calls are used by the fabric and the application code modules to notify each other of initialization, completion, and to acquire or change fabric parameters. We also added the modeling of a line card which mates 16 gigabit ethernet ports into the Avici backplane.

Our interface layer also handles the details of segmentation and reassembly so that an application is insulated from the transport details. This layer ensures that no packet is dropped by keeping lists of in-flight messages, which also aids in the generation of statistics.

3.3 Myrinet simulator

The myrinet simulator [6] was initially developed by Chen-Chi Kuo as a graduate student in the Computer Science Department at the University of Utah to be used in their full system simulator. We adapted just the myrinet part of the simulator, and added an event handling mechanism along with the packet tracking and upper layer frameworks written for the Avici simulator interface.

We generated hardware parameters from Myricom's documentation or by performing empirical tests on our cluster. Our connectivity topology is a two-dimensional (wrapped) torus of switches, with eight nodes to a switch, and was chosen for its scalability properties over more complex topologies. The links between switches consist of two parallel cables, giving a doubled hop-to-hop bandwidth of 2.56 Gb/s.

The same application codes designed for use with the Avici simulator couple directly to the interface we wrote to communicate with Utah's flit-level myrinet simulator, and similar parsing tools can be used to deduce statistics from the output of simulation runs.

4 Parallel code algorithms

Accurate characterization of network performance is a complex task. Simple numbers such as minimum latency or maximum bandwidth are not sufficient metrics to enable cross-technology comparisons. We augment these basic numbers with results from computational core algorithms from real parallel codes in use at Sandia. Results from the tests are deferred until the following section.

A code entitled `token_pass` is our simplest test. It arranges the participating processors in a virtual loop than iterates the passing of a "token" around the loop a certain number of times. Each processor awaits a message from its neighbor to the left, then delays a bit to simulate processing time, then sends a message to its neighbor on the right. By changing the size of the token to be large, we can perform accurate bandwidth measurements. By setting the payload to zero, we find the minimum message latency. Since only two processors at a time are ever involved in a communication, there are no contention effects to filter out from the results.

The codes `fan_in` and `fan_out` are generated from the same source file with different `#define` settings, as they perform quite similar functions. The former simulates a global reduction whereby each processor sends a message to the "host" processor. The sends are staggered slightly to avoid odd synchronization effects in the switches, and to simulate real life in which it is impossible to do clock-synchronized sends on a distributed machine. This test is good for measuring performance degradation due to internal fabric blocking. In the reverse mode, `fan_out` has the host processor sending staggered messages to all the other processors. This tests the blocking effects in the other direction. Performance numbers from `fan_in` and `fan_out` model the startup and shutdown events of parallel applications, which often include global broadcast and reduction phase.

The code `mesh` simulates a computational kernel from a two-dimensional finite element calculation. This class of structured grid codes is very common among the large scale calculations being performed today at the laboratories. The processors are laid out in a virtual two-dimensional mesh, and each processor will communicate with its immediate neighbors in both the x and y directions. The code performs a number of iterations of computation and communication cycles, which represents the real code's explicit time stepping algorithm as it solves a generalized partial differential equation.

We have made some modifications to the `mesh` code to model a torus topology, which is necessary for a code simulating periodic boundary conditions and arises in calculations on a spherical domain or in free space, for instance. In the toroidal topology, each processor always has four neighbors, unlike in the mesh where edge and corner processors have fewer neighbors.

5 Results

Our results are presented in order of the algorithms we used to test the networks, followed by a summary of all the tests.

5.1 Technology characterization

We ran the `token_pass` code with a one-byte payload to determine the minimum message latency between neighboring nodes in a 256-member virtual ring for all three

	`fan_in`				`fan_out`			
	Min	Avg	Max	σ^2	Min	Avg	Max	σ^2
Myrinet	13.19	1663.51	3313.83	956.55	13.43	1666.60	3319.09	957.94
Avici GigE	538.26	2941.32	4349.76	1055.05	63.24	159.54	258.60	45.13
Conventional GigE	24.83	2488.14	4345.48	1343.83	24.83	46.17	54.76	11.13
Avici fabric	4.68	129.72	233.49	66.55	21.18	141.44	233.43	61.62

Figure 2. Latency statistics for `fan_in` and `fan_out` codes. Units are in μs.

technologies. As mentioned earlier, since only two processors at a time are involved in communication, there are no contention effects. We compiled our results and listed the minimum, maximum, average, and standard deviation values for each study in Figure 1. As shown, the myrinet technology delivered very good latency and jitter (latency variation). Jitter in the absence of congestion is a function of network topology; it reflects the difference in distance between `token_pass` neighbors in the network. The raw fabric speed for the Avici switch is shown as the last line in Figure 1.

	Min	Avg	Max	σ^2
Myrinet	0.388	0.427	0.869	0.093
Avici GigE	1.380	1.386	1.530	0.024
Conventional GigE	1.564	1.595	3.532	0.244
Avici fabric	0.180	0.186	0.330	0.024

Figure 1. Minimum message latency results, in μs.

As a result of the Avici's higher fabric speed and path diversity in the 3D-torus topology, myrinet's performance is inferior to that of the Avici, as we configured a 2D torus for myrinet due to its physical constraints. As expected, since the Avici gigabit ethernet routes its packet through the Avici fabric, it had inherited the fabric's low jitter. The increases in its latency amounts to the sum of two transmission delays, when the packet arrives at the input and when it reaches the output of the Avici gigabit ethernet line cards. Store-and-forward switching is necessary here in order to prevent buffer underrun at the outgoing line card switch due to the large speed mismatch; a fabric link is 20 times that of the gigabit ethernet speed. Furthermore, because the ethernet standard imposes a minimum packet size of 64-byte, the original one-byte message was padded before transmission. Therefore, each of the transmission delays is actually $64 * 8/1000 = 0.512$ μs. Two times this value is roughly the increase seen in comparing to the fabric's latency.

The existing Opnet switch model does not emulate processing delay; consequently, the latency values that we obtained through simulation (Figure 1 row 3) are better than measured statistics. In our star topology, a packet will traverse either one or three hops depending on whether the immediate neighbor is on the same switch or not. Since switches today typically incur about 10 μs of processing delay, the values listed in Figure 1 would have an additional latency of 10 μs at the lower bound, and 30 μs at the upper bound, making the performance of a conventional switch the least favorable of the three.

Using `token_pass` and a 15 MB message, we measured throughput for each technology to verify the correctness of our simulation code. We chose that message size because it is large enough to fill the end-to-end communication pipe, a criterion necessary for throughput measurements. The end-to-end communication pipe is the product of the theoretical bandwidth and the round-trip time. The simulation throughput values are different from the corresponding theoretical bandwidth by less than half a percent.

5.2 Fan-in and Fan-out

Figure 2 lists the maximum, minimum, average, and standard deviation latency results for a 2 kB message from our `fan_in` study. The results for smaller messages go linearly to zero, and are not shown. The simulations had 255 sources each sending one messages to a single destination, thereby causing contention at the destination host machine. Myrinet performance is roughly 20% better than both the conventional and the Avici gigabit ethernet, because it has an effective bandwidth of 1.28 Gb/s to the destination host as opposed to the 0.98 Gb/s of gigabit ethernet.

Figure 2 also lists the latency results for a 2 kB message in the `fan_out` studies. A message was broadcast from the source to all destinations. As shown in the table, the conventional gigabit ethernet switches offer the best end-to-end latency by far, because these switches implement multicast in hardware, where a multicast packet is referenced and sent simultaneously to all multicast members. On the other hand, wormhole routing emulates multicast in software; this mechanism requires a source host to send a multicast packet multiple times, one for each multicast destination. Therefore, myrinet exhibited the worst latency performance. The Avici fabric fared better because of its higher aggregate bandwidth and diverse paths.

	Avici				Myrinet				Ethernet			
Size	Min	Avg	Max	σ^2	Min	Avg	Max	σ^2	Min	Avg	Max	σ^2
32	1.38	2.57	5.28	0.87	0.58	1.57	5.74	0.90	1.56	49.55	174.77	19.82
64	1.65	3.27	7.11	1.10	0.78	2.23	9.85	1.44	1.85	49.35	175.06	19.90
128	2.76	5.34	9.72	1.75	1.18	3.74	17.58	2.54	2.88	49.30	190.47	20.65
256	4.86	9.18	17.37	3.02	1.98	7.85	55.75	7.07	4.92	48.18	214.20	26.61
512	9.09	16.97	30.93	5.44	3.58	16.73	226.71	18.83	9.02	48.32	333.28	52.31
1024	17.49	32.56	55.95	10.25	6.78	36.54	441.90	43.75	17.21	84.04	586.44	98.29
2048	22.20	59.20	112.59	20.21	13.18	75.89	906.86	93.33	29.52	151.36	1088.28	188.40

Figure 3. Latency simulation results from `mesh` algorithm. Size is in bytes, all other fields are in μs.

5.3 Mesh and torus

The results for both the `mesh` and `torus` algorithms are included in this section as the codes are identical save for the extra edge connections in `torus`. Similar results will also be seen in both.

Message latency

The first data we present is a message latency. All messages sent (and received) are recorded with timestamps across all the iterations of the algorithm. The data in Figure 3 list the average time for a message to pass through the respective network, along with the standard deviation of the measurements and the maximum and minimum times. It is seen by comparing the average and maximum values for each technology that the maximum message transfer time can be up to an order of magnitude more than the average in the case of myrinet and conventional gigabit ethernet due to the presence of link contention. Only with the Avici switch are the numbers more comparable. These maximum numbers tend to pull up the averages.

The average transfer times for both the Avici and for myrinet are seen to be similar, while the conventional ethernet is larger due to the bottleneck at the second-stage switch in the center of the topology. It would be reasonable to use trunked links from the first-stage switches to the central switch to provide improved bandwidth and alleviate the bottleneck, but this type of scalability will not go too far as commodity switch vendors only provide small numbers of ports per switch. A fat tree using multiple switches is a possible solution, but expensive, and may also not reach high node counts as the switches directly connected to hosts will run out of ports in that case.

Up to a message size of 256 bytes, the myrinet network delivers average latencies about 1 μs lower than does the Avici ethernet. The y-axis intercept of the myrinet average line is about 1 μs while that for the Avici is 2 μs. This is due to the latency induced in the Avici switch core itself, which we measure to be the same amount. Myrinet switches add about 300 ns per hop, with an average of 3 hops per route in a 8×4 two-dimensional torus, to give the y-intercept seen there.

In the large message extreme, a 2 kB packet takes on average 59 μs to transfer through the Avici ethernet network, or 76 μs to transfer through the myrinet network; however, the worse case transfer time is a factor of eight greater for the myrinet, almost as bad as in the conventional gigabit ethernet network. Note that these are not raw transfer times, but the result of the interactions with transfers between other pairs of nodes on the network. This leads us to conclude that the effect is from the blocking induced by obstructing messages in the network traffic. The Avici switch is configured to be non-blocking by its extreme path redundancy and the fact that we do not overload the ports on each line card, so any difference between the maximum message transfer time and the average is due to output port contention, *i.e.,* when multiple messages are waiting to enter a single destination host. In the case of the conventional gigabit ethernet, messages may be blocked at the output ports of each of the up to three switches in the path from the source to the destination. The case for Myrinet involves up to six switches, but the bottleneck is not as great as in ethernet due to the multiple routes of the switches.

The message latency values for the related algorithm, `torus`, are well inside of one standard deviation away from those presented for `mesh`, and offer no solid conclusions. The algorithmic difference is that slightly more communication is occurring, and it is becoming more regular in that each processor talks to exactly four neighbors in `torus`. The underlying network is identical in both algorithms. This regularity seemed to help myrinet to provide fairer bandwidth sharing during contention, where results show a decrease in average and maximum latencies and standard deviations for all message sizes. This phenomenon is absent in the Avici case because of its much higher internal fabric speed. Conventional gigabit ethernet switches lack path diversity, and thus the increased offered load presented by the `torus` algorithm increased the queue depths at the out-

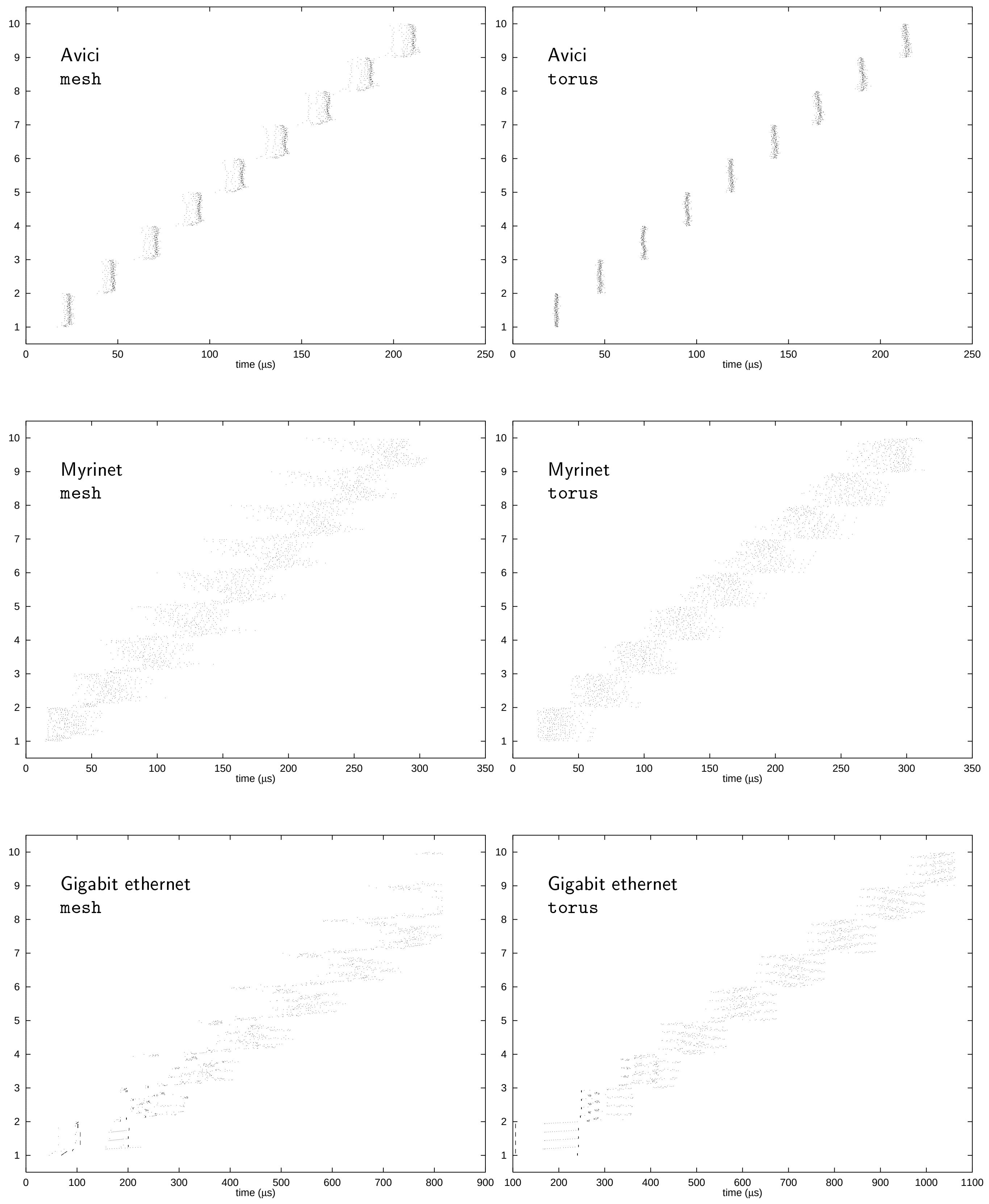
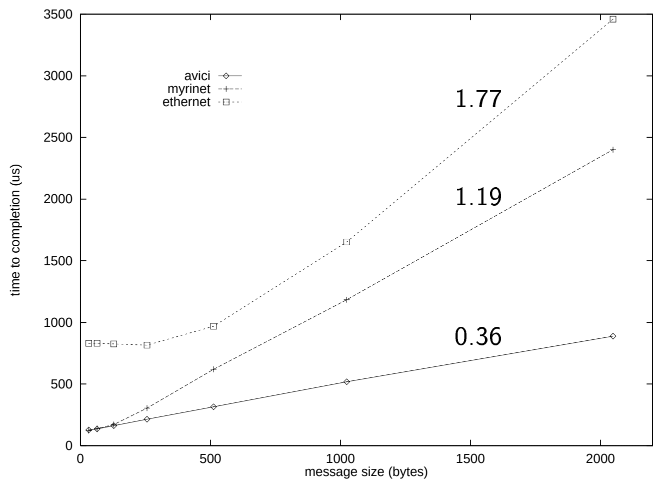


Figure 4. Completion times, 256 byte messages, for the three technologies and the two algorithms.



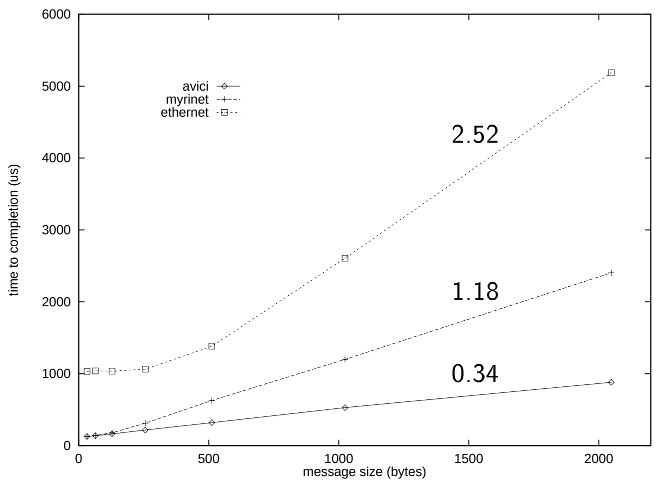


Figure 5. Total time for completion, `mesh` and `torus` topologies, with slope of linear fit.

put ports of the switches.

Completion times

The second data analysis we perform takes into account more of the details of the algorithm. Figure 4 shows plots of the results at a 256 byte message size, for each of the three technologies, and for both of the algorithms.

Each plot shows, for each iteration, and for each processor, the time when that processor completed that iteration. The unlabeled vertical axis is the iteration number of the algorithm, from 1 to 10. The horizontal axis is the global time, in microseconds, and varies from plot to plot as the completion times are quite different with respect to both message size and to network technology.

Each integral band of y-axis is broken up into 256 points, one for each processor, and a dot is placed in a processor's strip in a given iteration number at the time that processor has sent and received all messages necessary to proceed with the calculation of that timestep, or equivalently, when the processor has received the results of the previous iteration from all its neighbors.

One thing to notice in the plots is that some processors always complete much earlier than the others. For the `mesh` case, these are usually the ones on the corner which have fewer messages to exchange with their neighbors, as there are fewer neighbors. In the `torus` case, this is not true, and the individual bunches of dots tend to be more even, as the corner and edge processors can not advance too far ahead of the rest of the fray in the middle.

At the relatively small 256 byte message size shown in the figure, the iteration bunches are well separated from each other as most of the time to completion of each iteration is taken up by computation time, represented in our simulation by a sleep of 10 μs. An ideal network which used no time to transfer messages would show perfectly vertical lines at each iteration, with the last line (between 9 and 10) at 90 μs. Anything more than this is the effect of waiting for communications.

Unshown are the plots for the rest of the studied range of message sizes, where looking at just the Avici results, we observe that the total time to completion is gradually increasing, from 140 μs for 32 byte messages, to 900 μs at 2 kilobyte messages, and that each iteration bunch is becoming separated into individual stripes, with the edge processors finishing earlier than the bulk in the center. For the `torus` case there is no obvious striping.

In the myrinet network case, where the groups are fuzzier as the effect of the larger maximum communication times listed in Figure 3. The apparent patterns in the large message size plots show the discrepancy in transfer time between nearby nodes and distant nodes in the mesh (or torus) as messages sent farther through the network are subject to more potential points of blocking. Total time to completion for these simulations are the same as for Avici at small message sizes, to about three times longer in the large message case.

The results for the conventional gigabit ethernet cascade of switches feature x-axis ranges consistently three to six times larger than those for the Avici plots. Great multi-millisecond stripes can be seen in the large message size plots for the ethernet where whole regions of the two dimensional mesh proceed into later iterations while other regions are still working on the communications associated with earlier iterations. In the `torus` case this horizontal striping is more pronounced but the iterations are forced to be more temporally bunched as the added toroidal communication patterns introduce more dependencies between processors.

This spread in iteration number is able to occur since there is no global synchronization step between iterations—each processor is permitted to proceed to the next iteration as long as it has received results from the previous iteration from all its neighbors. Following this thought, it can be seen that a certain processor can get up to two iterations in time away from those processors which are neighbors of its immediate neighbors. This continues up to the boundaries of the mesh, which for our 16×16 case means that the spread can proceed up to eight iterations apart.

It is interesting to notice the rate of degradation of network performance with increasing message size. This is shown in Figure 5 for the two algorithms. A linear fit of the largest two points for each technology and each algorithm is overlaid on the plot near the corresponding curve. This slope represents the scalability of the given network technology to the two algorithms under increasing message size. Both codes give the same scalability performance on the first two networks, but for conventional gigabit ethernet, the increased offered load seen in the `torus` algorithm renders the network less scalable as message sizes grow.

Summary

The average message latency delivered by the three network technologies is affected both by available bandwidth and the presence of bottlenecks. The conventional cascaded gigabit ethernet without trunking is hampered severely by both these factors. Fabric blocking is seen to be bad for parallel algorithms in that it increases the maximum latency seen by any particular message, and since all messages must eventually reach their destination before the code can complete, that maximum latency value is crucial to the wall-clock performance of a code. The Avici switch is seen to have the smallest amount of fabric blocking, while the myrinet fabric offers potential blocking points at every switch along the path of a message.

The algorithm we tested was chosen due to its ubiquity in parallel scientific computing. It emphasizes the nature of locality in many algorithms in production use today, but points out in the results above that not all communications will physically be local even though in the virtual topology they may appear to be nearest neighbor. This mapping of algorithmic topology to physical topology is crucial for application performance. We did not model an algorithm which involved a global synchronization, which are also common especially for those that do disk input and output. The effects of this communication pattern can be discerned by looking at the `fan_in` and `fan_out` results of Section 5.2.

6 Conclusions

We have presented the results of analysis of three different major network architectures for parallel commodity computing. It is important to choose the network correctly as it can have a large impact on all but the most embarrassingly parallel applications, and may be the source of up to half of the cost of the entire machine. Important factors to consider are raw performance figures such as bandwidth and latency, as well as more complex parameters such as jitter, routing, multicast support, and distribution of blocking in the fabric.

Since our network design goal is to facilitate the performance of real applications, we evaluated the performance of the three network technologies when applied to specific application cores important to our users. In this context we analyzed timing results gathered from the networks and drew conclusions from our knowledge of the network about its effect on performance of the application.

Our simulation results show that myrinet behaves well in the absence of congestion. Under heavy load, its latency suffers due to blocking in wormhole routing. Also myrinet is limited from scaling too far due to the short cable length problem. Future development by Myricom may alleviate that constraint, although the cost to latency or budgets is unknown. The simplicity in the myrinet switch results in low per-connection cost; however, the non-commodity nature of the host network interface cards keep that side of the connection expensive.

Conventional gigabit ethernet switches can not scale to support more than 64 gigabit ethernet ports, which lead to the introduction of a topology which involves cascading multiple stages of small switches. The presence of multiple hops in a path between hosts, and the store-and-forward nature of legacy ethernet leads to unacceptable message delays. Bandwidth bottlenecks at the topmost switch in the cascade are also a problem.

The Avici terabit switch router has an internal fabric which is quite similar to myrinet, in that it is a very high-bandwidth three-dimensional torus using source routing and simple non-buffering switches. The line cards present standard gigabit ethernet connections to hosts, though, in keeping with the current commodity favorite. Our simulations show that the Avici switch outperformed myrinet on large messages (above 512 bytes), and was comparable in the small-message regime. From a cost standpoint, Avici is only slightly cheaper than myrinet for a comparable topology, and is expected to reduce in cost with further penetration of gigabit ethernet into the market.

References

[1] Dally, W. “Scalable Switching Fabrics for Internet Routers.” Computer Systems Laboratory, Stanford University and Avici Systems. July 1999.

[2] Duato, J., Yalmanchili, S., and Ni, L. “Interconnection Networks: an Engineering Approach.” *IEEE Computer Society Press.* 1997. pp. 11–16.

[3] Held, G. *Ethernet networks: design, implementation, operation, management.* John Wiley & Sons, Inc. 1998, pp. 78–95.

[4] Jacobsen, O. J., ed. “From the editor.” *The Internet Protocol Journal.* **2**, n. 3, pp. 1ff, 1999.

[5] King, A. “Terasim: the Simulator for Avici TSR.” Avici Systems, Inc., 1997.

[6] Kuo, C. C. “The Avalanche Myrinet Simulation Package.” Department of Computer Science. University of Utah. 1997.

[7] Seifert, R. *Gigabit Ethernet: technology and applications for high speed LANs*, Addison-Wesley, 1998, pp. 141–280.

[8] Stevens, W. *TCP/IP Illustrated.* Addison-Wesley, 1994–1996.

[9] Myricom. `http://www.myri.com/myrinet/overview/index.html`. 1999.

[10] High performance networking forum. `http://www.hnf.org`. 1998.

[11] MIL3. `http://www.mil3.com/products/modeler/home.html`. 1999.

[12] Giganet. `http://www.giganet.com`. 1999.